Document Object Model

U okviru lekcije upoznaćemo se sa manipulacijom Document Object Modela, odnosno rukovanjem HTML objektima kroz JavaScript. Obradićemo metode za manipulaciju DOM-a i videti rezultate na praktičnim primerima.

Tokom ove lekcije naučićemo:

- šta je to DOM;
- kako promeniti sadržaj HTML elementa;
- kako promeniti style/CSS HTML elementa;
- kako koristiti događaje i povezivati ih sa HTML elementima.

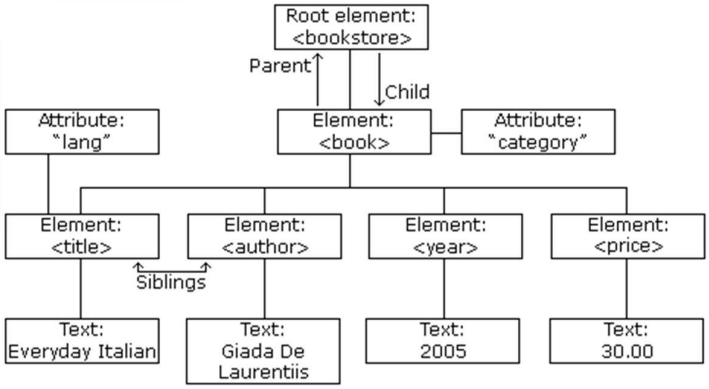
Šta je DOM?

DOM je skraćeno od *Document Object Model*. Može se reći da je to standard koji se koristi za dinamički pristup i izmenu svih elemenata jedne stranice. Ti standardi su podeljeni na tri dela:

1. Core DOM – standardni model za sve tipove dokumenta;
2. XML DOM – standardni model za XML dokumente;
3. HTML DOM – standardni model za HTML dokumente.

Core DOM definiše događaje i modele dokumenata koje web platforma koristi.

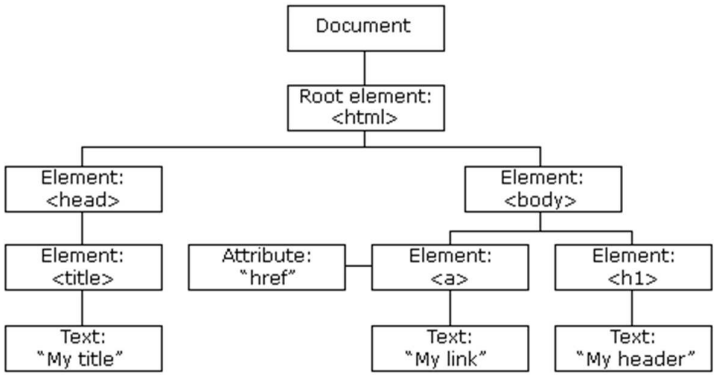
XML DOM definiše standard za pristup i izmene XML dokumenta. Na slici je prikazano stablo/drvo objekata na primeru biblioteke.



Slika 8.1. Drvo XML DOM objekta[1]

HTML DOM je sastavljen od niza objekta koji je prikazan kao stablo/drvo na sledećoj slici. Taj standard je definisan za HTML dokumente.

[1] https://www.w3schools.com/xml/dom_intro.asp



Slika 8.2. Drvo HTML DOM objekta[2]

Kako bi slike bile jasnije, navešćemo osnovne osobine strukture jednog stabla/drveta:

- na vrhu imamo *root* element, što se može prevesti kao *koren* jednog stabla;
- svi elementi osim korena imaju svog roditelja – element koji je iznad njega;
- element može imati više dece – to su oni elementi koji su ispod i može ih, dakle, biti više;
- svaki roditelj sa svojim potomkom gradi jednu granu;
- element koji se nalazi na dnu stabla/drveta se zove list.

Analizom ovakve strukture možemo zaključiti da su elementi međusobno povezani na osnovu određenih pravila i čine jedno drvo DOM objekta.

Promena sadržaja HTML dokumenta

Sada kad smo se upoznali sa strukturom DOM-a, proći ćemo kroz primer i izvršiti manipulaciju nad HTML dokumentom.

Najbitniji i osnovni objekat DOM strukture, koji ćemo često koristiti, jeste Document. Pristupamo mu korišćenjem document svojstva Window objekta.

Postoji više funkcija za selektovanje elementa korišćenjem document svojstva; njihov pregled nalazi se u tabeli 8.1.

Metoda	Opis
getElementById(id)	Ovom metodom dobavljamo element preko id-ja.
getElementsByClassName(name)	Ovom metodom dobavljamo sve elemente preko naziva klase
getElementsByTagName(name)	Ovom metodom dobavljamo sve elemente preko naziva HTML taga
querySelector(cssSelector)	Ovom metodom dobavljamo element za prvi css selektor
querySelectorAll(cssSelector)	Ovom metodom dobavljamo sve elemente za dati css selektor

Tabela 8.1. Metode document objekta

[2] https://www.w3schools.com/whatis/whatis_htmldom.asp

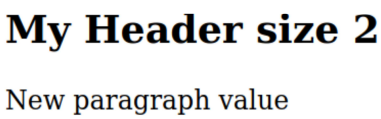
Sledeće što nam je potrebno kako bismo izmenili sadržaj, pored selektovanja elementa, jeste innerHTML svojstvo. Korišćenjem ovog svojstva dobijamo mogućnost upisa ili izmene vrednosti elementa selektovanog nekom od metoda prikazanih u tabeli.

U sledećem primeru koristićemo metodu getElementById(id) i putem innerHTML svojstva ćemo izmeniti tekst na paragrafu koji ima id p-id.

Primer 1

```
<!DOCTYPE html>
<html>
<head>
          <title>My Page</title>
</head>
<body>
         <h2>My Header size 2</h2>
         <p id="p-id">Paragraph value</p>
         <script>
                  document.getElementById("p-id").innerHTML = "New
paragraph value";
         </script>
</body>
</html>
```

Rezultat na stranici, kada je pokrenemo unutar pregledača, izgledaće ovako:



Slika 8.3. Prikaz rezultata primera 1

Izmenite kod u radnom okruženju tako što ćete ispisati svoje ime u naslovu veličine h4 sa definisanim id-om `ime`, prezime u paragrafu i izmeniti prvi naslov tako da ispisuje Vaše ime i prezime pomoću JavaScripta odn. jedne od getElement metoda.

Dodavanje klase HTML elementu

U ovom poglavlju ćemo postaviti jedan <p> element i manipulisati njegovim stilom.

Kako bismo mogli da manipulišemo stilom, pre svega, potrebno je definisati taj stil. U ovom slučaju stil će biti definisan u <head> delu, u <style> elementu, koji nam služi za unos CSS koda. Tu ćemo kreirati pstyle klasu.

Kao i u prethodnom primeru, koristićemo metodu getElementById(id), kojoj prosleđujemo id elementa <p>. Nakon toga upotrebljavamo novo svojstvo za promenu, tj. dodavanje CSS klase className.

Primer 2

```
<!DOCTYPE html>
<html>
<head>
          <title>My Page</title>
          <style>
                    .pstyle {
                              width: 400px;
                              height: auto;
                              background-color: #066c94;
                              text-align: center;
                              color: white;
                    }
          </style>
</head>
<body>
         <h2>My Header size 2</h2>
         <p id="p-id">Paragraph value</p>
         <script>
                  document.getElementById("p-id").className =
"pstyle";
         </script>
</body>
</html>
```

Rezultat na internet stranici će izgledati ovako:

Slika 8.4. Prikaz rezultata primera 2

Izmenite kod u radnom okruženju tako da naslov h3 sa definisanim id-om ‘crveni‘ koji ispisuje Vaše ime bude crven, sa zelenim slovima, širine i visine 300px, centriran ulevo. Napravite paragraf koji ispisuje Vaše prezime i pomoću JavaScripta mu dodelite id ‘zeleni‘ gde ćete pre toga definisati id ‘zeleni‘ unutar css tako da pozadina bude zelena, sa crvenim slovima, širine i visine 300px i centriranje udesno.

Pitanje

Kojom metodom selektujemo element preko imena klase?

- getElementById()
- **getElementsByClassName()**
- getElementsByTagName()
- querySelector()
- querySelectorAll()

Objašnjenje:

Tačan odgovor je getElementsByClassName() metoda. Kao ulazni parametar metodi prosleđujemo naziv klase, a kao rezultat dobijamo niz sa svim elementima koji imaju prosleđenu klasu.

Pridruživanje događaja HTML elementu

Nakon prethodnih segmenata smo spremni da zaronimo dublje u moć DOM objekta i kroz primer naučimo da putem događaja (eventa) promenimo tekst u <p> elementu.

Reč *event* znači *događaj*, što bi u našem slučaju značilo da nakon nekog događaja nešto treba uraditi. Lista svih događaja se može videti na sledećem linku. Postoje dva načina za definisanje događaja: jedan je unutar samog JavaScript koda, a drugi je unutar HTML elementa. U našem primeru, događaj ćemo definisati unutar HTML elementa kao atribut i to će biti događaj onlick. Vrednost ovog atributa će biti funkcija koja će se pozvati klikom miša na pomenuti HTML element, odnosno, u našem primeru – klikom na dugme.

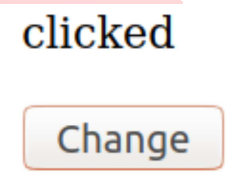
Događaj onclick poziva funkciju clickFunction(), koja sadrži već poznati kod koji će promeniti tekst <p> elementa.

Primer 3

```
<!DOCTYPE html>
<html>
<head>
          <title>My Page</title>
</head>
<body>
         <p id="p-id">Click the button...</p>
         <button onclick="clickFunction()">Change</button>
         <script>
                  function clickFunction(){
                           document.getElementById("p-id").innerHTML =
"clicked";
                  }

         </script>
</body>
</html>
```

Rezultat na stranici, kada je pokrenemo putem pregledača, izgledaće ovako:



Slika 8.5. Prikaz rezultata primera 3

Izmenite kod u radnom okruženju tako da kada se klikne na dugme paragraf menja boju slova u zeleno odn. dodelite mu klasu koji unutar css-a definiše zelena slova.

Radno okruženje

Rezime

- DOM je skraćenica od Document Object Model; ovaj model se koristi za dinamički pristup i izmene svih elemenata jedne stranice.
- Metode koje koristimo da bismo dinamički menjali elemente na stranici su: getElementById(), getElementByClassName(), getElementsByTagName(), querySelector() i querySelectorAll().
- Svojstvo innerHTML je svojstvo koje koristimo da bismo izmenili tekst nekog elementa.
- Svojstvo className je svojstvo koje koristimo da bismo izmenili stil, odnosno dodali klasu elementu.